Model-based Design and Testing of Decisional Autonomy and Cooperation in Cyber-physical Systems

Francesca Saglietti, David Föhrweiser, Stefan Winzinger, Raimar Lill
Software Engineering (Informatik 11)
University of Erlangen-Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
saglietti@informatik.uni-erlangen.de, david.foehrweiser@informatik.uni-erlangen.de,
sten.winzinger@fau.de, raimar.lill@informatik.uni-erlangen.de

Abstract—This article presents a study on the benefits offered by Coloured Petri Nets in capturing and separating permanent and temporary behavioural information and on the systematic support they hereby provide to model-based design and testing of cyber-physical systems. In particular, it illustrates the application of CPN modelling to capture the behaviour of cooperative mobile robots and highlights their benefits in terms of compactness and scalability. Finally, the article reports on the applicability of test case generation algorithms supporting the coverage of the underlying CPN models with respect to different testing criteria.


I. INTRODUCTION

In recent times, cyber-physical systems increasingly tend to involve and to rely on the cooperative interplay of autonomous agents [1]. Evidently, the intention is to achieve, in addition to the individual performances offered by the individual agents, further beneficial add-on values “emerging” from their cooperation. Typically, emergent behaviour may concern

- an increase in functionality or performance by means of the coincidental contribution of more agents to a common action (e.g. the seizing of particularly voluminous objects or the lifting of particularly heavy objects), or
- a consistent co-existence of several agents in a common working area by timely detection and resolution of potential conflicts (e.g. avoidance of collisions).

On the other hand, emergent behaviour may also involve unexpected and undesired behavioural patterns resulting from exceptional interactions between autonomous systems which could not be anticipated during development or analysis.

In application areas involving ultra-high reliability requirements, as is the case for safety-relevant tasks, malicious emergent behaviour must be excluded before operation by accurate analysis techniques.

A typical cyber-physical domain increasingly involving the cooperative paradigm concerns robotic applications, where autonomous agents possibly originating from different suppliers are applied within a common application environment. Cooperating robotic agents may have been individually developed and tested to achieve similar functionalities concerning their sensing, perception, reasoning and acting capabilities. Depending on their origin, they may differ a. o. with respect to their performance, consumption of resources, reliability and reconfiguration capabilities.

In addition, behavioural variability usually also heavily depends on environmental conditions, e. g. meteorological conditions or ground properties possibly affecting the individual and the cooperative movement behaviour of agents.

In order to ensure operational safety, design and verification of cooperative cyber-physical systems must be conceived such as to address as systematically as possible the operational variants resulting from varying external circumstances. Model-based engineering provides a classical strategy to be adopted for this purpose. In order to support reusability and scalability, the modelling approach taken in the following relies on two different modelling elements:

- a basic, static part of the model refers to the generic representation of the cooperative concept planned, in particular addressing long term invariants like plant topology and global reconfiguration strategies;
- further dynamic model elements concern the variability of individual scenarios which depend on momentary properties of cooperating agents, missions and environmental constraints.

The rest of this article presents a study on the benefits offered by Coloured Petri Nets in providing such a notation and in supporting structural model-based testing based on coverage criteria to be fulfilled by automatic test case generation.
II. COLOURED PETRI NETS


Like for common Petri Nets, the current net state is encoded by the token marking, where each CPN place is characterized by a corresponding colour set specifying the type of tokens that may be allocated to that particular place.

The firing rules governing the CPN behaviour involve additional net information:

- **arc expressions** indicating multi-sets of colours,
- **transition guards** indicating predicates.

For a given transition, a variable binding denotes an assignment of variables occurring in input arc expression with colours from colour sets of corresponding input places. Particular variable bindings enable the firing of transitions if

- the transition guards are fulfilled and
- for each input place and each colour the number of tokens of that colour is at least as high as the number indicated by the corresponding input arc expression.

The transition firing yields

- the removal from each input place of as many tokens of each colour as indicated by the corresponding input arc expression and
- the addition to each output place of as many tokens of each colour as the corresponding output arc expression indicates.

Apart from providing valuable analysis and simulation techniques and tools supporting full expressive power [2], CPN explicitly supports the splitting of generic and scenario-dependent information mentioned in section I by

- encoding permanent, scenario-independent information in the static CPN graphical layout, while
- encapsulating time-varying, scenario-dependent information within the type-specific, dynamic marking.

In more detail, CPN elements can be used to capture the following information:

- **CPN transitions** to encode generic actions;
- **CPN token markings** (i.e. states) to encode contextualized temporary conditions;
- **CPN events** to encode action instances;
- **CPN state pairs** to encode contextualized actions;
- **CPN state sequences** to encode contextualized operational scenarios.

III. EXAMPLE: MOBILE ROBOTS

The example considered in this article concerns a mobile robotic application arisen in the context of a European cooperation. It addresses the movement of an arbitrary number of agents along an arbitrarily long linear path subdivided into segments annotated by consecutive numbers.

Each robot starts its mission by taking a new order consisting in reaching a given target segment as autonomously as possible without colliding with other vehicles or further objects.

To do so, each robot first determines the direction in which to proceed depending on its current position: to the right if the number of the target segment is higher than the number of its initial location, to the left otherwise.

Successively, the robot scans the next segment to be accessed in order to find out whether this is physically possible. In case the segment is not free, the vehicle must be able to distinguish between

- a **passive obstacle** where the target segment is occupied by some inactive object; in this case the robot raises an alarm initiating an operation shutdown;
- a **traffic jam** where the target segment is occupied by a robot moving or trying to move in the same direction as the observing robot; in such a case the robot retries a given number of times to access the segment; if unsuccessful, it raises an alarm initiating an operation shutdown;
- a **contra-flow** where two robots are facing each other, each aiming at moving towards the other one: in such a case the robots resolve their conflict by swapping their positions.

Finally, once the target segment is reached, the robot concludes its mission and waits for the indication of a new target.

Evidently, this example involves only a very limited number (namely 6) of generic action categories carried out by each agent, namely

- to take an order (action **start**);
- to complete an order (action **finish**);
- to move one step to the right (action **move right**);
- to move one step to the left (action **move left**);
- to exchange the position with another robot (action **swap positions**);
- to identify (at least part of) a robot queue which does not progress in spite of a given number of access trials (action **detect total gridlock**).

The overall behavioural complexity, however, results from the combinatorial multiplicity of the instantiation of these action patterns in the light of the individual capabilities of the participating agents under varying operational conditions.
Such variability includes a.

- the momentary number of active robots in the warehouse factory;
- the robot-specific capabilities depending on their individual performance characteristics and on their present location;
- the environmental context of an action including all further agents which, although not directly involved in the action, may indirectly influence its outcome.

The combinatorial multiplicity of such aspects asks for a formal instrument allowing to encode it during design and to analyse it during testing. In order to illustrate the expressive power, compactness and scalability of CPN, they will be applied to the robot warehouse just introduced.

The 6 actions mentioned above (start, finish, move right, move left, swap positions and detect total gridlock) are modelled by corresponding CPN transitions, while the information required to activate these actions and to record their effects are stored in type-specific tokens marking 6 CPN places:

- tokens in CPN place new missions represent tasks to be allocated to given robots;
- tokens in CPN place ongoing missions represent tasks already allocated to given robots and not yet concluded;
- tokens in CPN place locations represent the positioning of robots and of any passive objects;
- tokens in CPN place access trials represent potential waiting queues attempting access;
- tokens in CPN place factory shutdown represent the raising of an alarm interrupting operation;
- tokens in CPN place missions concluded represent a list consisting of all missions meanwhile successfully carried out.

The 6 transitions and the 6 places identified can be taken to provide the static net structure shown in Figure 1 which builds the graphical basis of the CPN model. For all other modelling details (including arc expressions and transition guards) the reader is kindly referred to [5]. As envisaged, the CPN model introduced allows to split between

- generic, logistics-related concepts captured by an extremely compact net structure limited to only 12 nodes (6 transitions and 6 places) and
- specific, varying operational modes captured by transition firings with respect to operational data, i.e. by CPN events.

As expected, the number of CPN events grows considerably with the number of mobile robots involved with operation. For example, assuming the following initial scenario where \( r \) robots were placed as uniformly as possible at the ends of a lane with \( s \) segments (with \( r \leq s \))

\[
\text{start (i)} = \begin{cases} 
  i & \text{if } 1 \leq i \leq \left\lfloor \frac{r}{2} \right\rfloor \\
  s + i - r & \text{otherwise}
\end{cases}
\]

while their targets were chosen such as to maximize the paths to be traversed by the robots:

\[
\text{target (i)} = \begin{cases} 
  s + i - \left\lfloor \frac{r}{2} \right\rfloor & \text{if } 1 \leq i \leq \left\lfloor \frac{r}{2} \right\rfloor \\
  i - \left\lfloor \frac{r}{2} \right\rfloor & \text{otherwise}
\end{cases}
\]

In such a case, the number of CPN events grows from 384 (in case of 4 robots moving along 10 segments) to about twice as much, namely to 763 (in case of 5 robots moving along the same path).

Figure 2 shows how CPN events grow in size with number of robots and segments.
Even more dramatic is the increase for CPN states and CPN state pairs, where the two examples just considered show a growth from 39910 to 361688 CPN states and from 103047 to 968206 CPN state pairs. Figure 3 (resp. Figure 4) illustrate the increase of CPN states (resp. CPN state pairs) with increasing number of robots and segments.

**Fig. 3.** Number of CPN states vs. number of robots and number of segments

**Fig. 4.** Number of CPN state pairs vs. number of robots and number of segments

### IV. Model-Based Testing

In addition to a compact and scalable representation supporting the instantiation of varying operational scenarios, the modelling approach suggested also provides for systematic testing coverage measures and thus for objectively reproducible testing targets. Among the testing coverage criteria identified in [6] are the following ones:

- the *transition coverage* criterion which requires each CPN transition to be fired during testing;
- the *event coverage* criterion which requires each CPN event to occur during testing;
- the *state coverage* criterion which requires each CPN state to be reached during testing;
- the *state pair coverage* criterion which requires each possible pair of consecutive CPN states to be reached in succession during testing.

Transition, event and state pair coverage criteria can be hierarchically organized such as to reflect the increasing rigor they involve as well as the increasing effort they require. In fact, while transition coverage testing addresses only the observation of generic actions (e.g. the generic movement of a robot) regardless of their operational context, event coverage extends this behavioural scope by capturing also the potential multiplicity of action instances (e.g. the specific movement of a robot on particular plant segments). Finally, state pair coverage goes even further by requiring to take into account all possible operational pre- and post-conditions of each event, i.e. to distinguish between events occurring under different global conditions (e.g. the same swapping of position of two robots assuming different positions of further robots in the area).

The automatic generation of test cases fulfilling CPN-based coverage criteria can be achieved by different strategies. *Analytic approaches* based on state exploration and graph search algorithms assume finite state spaces, fully explorable within affordable time, memory and computing capacity limits, on which to pursue systematic coverage maximization. Among the best performing algorithms inspired by [7] is the so-called *hot spot prioritization-based algorithm* consisting in favouring the inclusion of those test cases covering as many yet uncovered CPN entities as possible. This particular algorithm assumes the absence of cycles in the state graph, a condition which is fulfilled for the robot factory scenario introduced above, resulting in test case sets of sizes shown in Table I.

**TABLE I. RESULTS BY HOT SPOT PRIORITIZATION**

<table>
<thead>
<tr>
<th>Coverage Criterion</th>
<th># CPN Entities to be Covered</th>
<th># Test Cases Achieving Full Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>transition coverage</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>event coverage</td>
<td>384</td>
<td>59</td>
</tr>
<tr>
<td>state coverage</td>
<td>39910</td>
<td>6369</td>
</tr>
<tr>
<td>state pair coverage</td>
<td>103047</td>
<td>18644</td>
</tr>
</tbody>
</table>

Among the advantages offered by this class of search approaches is the guarantee of termination and the a priori knowledge of the effort required by searching. Among the limitations are the already mentioned assumption about state finiteness and the fact that during search they do not explicitly
target a global minimization of the number of test cases required.

Evolutionary approaches [8], [9] optimize test case generation by genetically manipulating and evaluating successive generations of individuals, where each individual consists of a number of test cases. Genetic manipulation is carried out by random selection, recombination and mutation of individuals, while evaluation is based on a fitness function reflecting the degree of fulfillment of the underlying optimization goals. By weighting the priority of different optimization targets, the approach can serve to pursue multi-objective optimization, e. g. both coverage maximization and test amount minimization. In spite of these advantages, for the example considered they were outperformed by hot-spot prioritization.

On the other hand, as evolutionary techniques do not require the explicit representation of the state graph and can thus dynamically extend the search space, they allow for infinite state spaces. In addition, in case the fulfillment of complete coverage requires prohibitively high amounts of testing, they support at least the identification of optimal test case sets of given size, i.e. the determination of a given number of test cases capable of maximizing the relative coverage achieved. This can be achieved by extracting Pareto-optimal solutions among all those generated by evolution, i.e. by extracting solutions not outranked by any other generated alternative with respect to both size and coverage. The generation of a Pareto front is helpful in that it permits the test manager to dynamically optimize the choice on test cases by taking into account a posteriori knowledge like affordable testing time left.

V. CONCLUSIONS

This article stresses the relevance of modelling the behaviour of cyber-physical systems by notations supporting the distinction between the following classes of information.

Basic behavioural rules addressing permanent, scenario-independent information can be statically encoded by graphical representations of global operational concepts including aspects like

- plant topology;
- route re-planning strategies.

Scenario-dependent information subject to environmental variance, on the other hand, can be encapsulated into dynamic entities subject to changes and addressing the time-dependent variability of operation including momentary information on

- mission properties (including aspects such as criticality, required capabilities, working areas involved, degradability);
- robot status (including aspects such as number, location, functionality, quality of performance, external resources available, decisional autonomy);
- environmental peculiarities (including aspects such as local complexity of perception / action, environmental anomalies requiring counter-measures).

Coloured Petri Nets were shown to support both the expressive power required to achieve this distinction and a well-founded basis on which to apply test generation procedures with objectively reproducible coverage targets.

Ongoing work is being devoted to the extension of the approach presented for the purpose of encoding more sophisticated reconfiguration patterns by means of type-specific tokens.

Future work will be dedicated to the extension of the testing procedures developed so far to target – in addition to the structural coverage of behavioural multiplicity – also the provision of operationally representative evidence as required for reliability evaluation.

ACKNOWLEDGEMENT

It is gratefully acknowledged that part of the work reported was sponsored by the German Federal Ministry of Education and Research (BMBF) within the European Union Research Programme ARTEMIS (Advanced Research and Technology for Embedded Intelligence and Systems), project R3-COP (Resilient Reasoning Robotic Co-operating Systems); in addition, part of the ongoing and future work mentioned at the end of the article is being carried out within the follow-up project R5-COP (Reconfigurable ROS-based Resilient Reasoning Robotic Co-operating Systems).

REFERENCES


